Building and Running Apps

When you want to run your app to test or debug it, you build it using the Xcode build system. If there are no build errors, you can run it in a simulator or on a device.

> **Note:** To ensure your app functions correctly on the iOS-based device models you expect the app's users to use, after simulating your app's execution in simulators, you must test it on corresponding devices.
>
> To run your app on a device, you must be a member of a development team; see "Becoming a Member of a Development Team" (page 9) for details. You must also identify to Xcode the devices you intend to use for development, as described in "Replacing an Expired Certificate" (page 15).

These are the general steps to build and run an app:

1. Specify the build-time environment.
2. Specify the destination for which you want the product to be built: a simulator or a device.
3. Build the app.
4. Run the app.

This chapter describes each of the steps required to run your app. Start with "Running Sample Apps" (page 23) if you're interested in seeing apps that showcase iOS features.

> **Content specifications:** This content is written for Xcode 4.3 and iOS SDK 5.0.

Running Sample Apps

The iOS Developer Library provides several resources that help you learn about the iOS app development process. One of these resource types is sample code. You can access sample-code projects in the Xcode documentation organizer or in your web browser by visiting the iOS Dev Center. You can run sample code in a simulator. If you're a member of a development team, you can also run sample code on devices. See "Becoming a Member of a Development Team" (page 9) for details.

To get sample code using Xcode:

1. Choose Window > Organizer to open the Xcode Organizer, and click Documentation to display the documentation organizer.
2. Click the Browse button (the one with the eye icon) in the navigator selector bar of the documentation organizer.
3. Select the library for which you want to view sample code.
4. In the text field in the content area, enter "sample code".
5. In the content area, click the name of the project you want to open.
6. In the sample-code project page, click Open Project.
7. Choose a location for the project.

To get sample code using your web browser:

1. In your web browser, go to http://developer.apple.com/devcenter/ios, and click the iOS Developer Library link.
2. In the list on the left, click Sample Code under the Resource Types group.
3. In the Documents list, click the name of the project you want to open.
4. Click Download Sample Code.

   An archive containing the project directory is downloaded to your Mac. The archive may be automatically expanded for you. If it isn't, expand the archive by double-clicking it.
5. Navigate to the sample-code project directory.
6. Double-click the project package, a file with the `.xcodeproj` suffix.

   For example, for the HelloWorld project, double-click `HelloWorld.xcodeproj`. This action opens the project in Xcode.

   Another way to open the project is to drag the project package to the Xcode icon in the Dock.

Troubleshooting:
- Xcode doesn't launch.

  Download Xcode and install it on your computer. To learn how, visit iOS Dev Center.

With the sample-code project open in Xcode, follow the instructions in the following sections to build and run the app.

The Build-and-Run Workflow

This section describes each of the steps in the build-and-run workflow.

Specifying the Buildtime Environment

When you build your app, Xcode uses a build environment made up of frameworks, libraries, apps, command-line tools, and other resources. Each revision of the iOS SDK makes improvements to this environment to, for example, add user-interface features or improve compilers and resource-processing tools. In addition to these resources, you can specify whether you want to build your app to debug it or to distribute it to customers. This section describes how to set your build-time environment.

Setting the SDK Used to Build Your App

One of the main factors that determine how Xcode builds your app is the SDK used to build it.

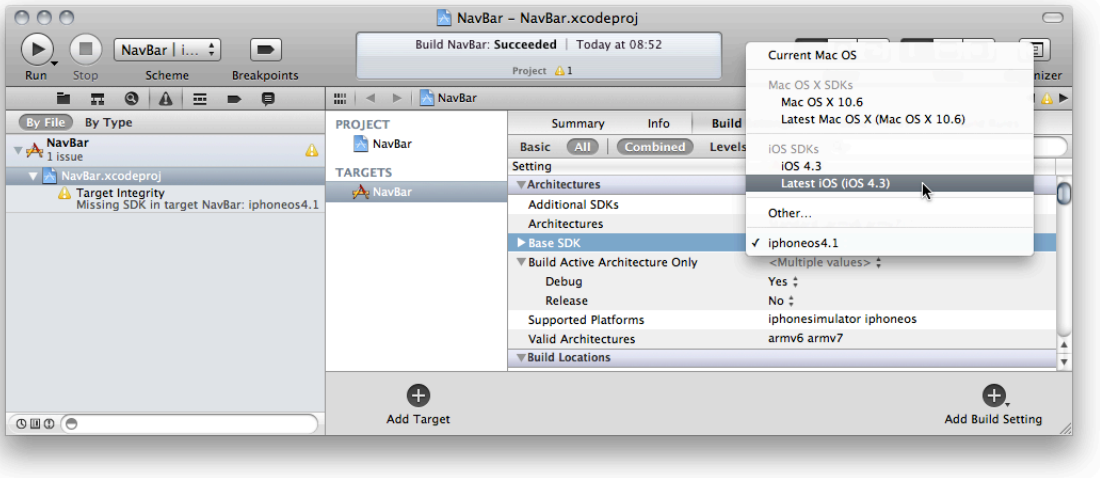
To specify the SDK to use to build your app, set the Base SDK build setting to the appropriate SDK.

Note: To ensure minimal reconfiguration of your projects as you adopt new SDK releases, instead of a specific SDK release, set the base SDK for your projects to Latest iOS. This way your project always uses the latest available SDK in the toolset.

Base SDK Missing

If your project has its Base SDK setting set to a particular iOS SDK release, when you open that project with a later Xcode toolset distribution in which that SDK release is not available, the Base SDK setting has no valid value. In this case, the issue navigator lists a Missing SDK issue, as shown in Figure 3-1. To fix the Missing SDK issue, set the base SDK for the target to an available SDK release or to Latest iOS.

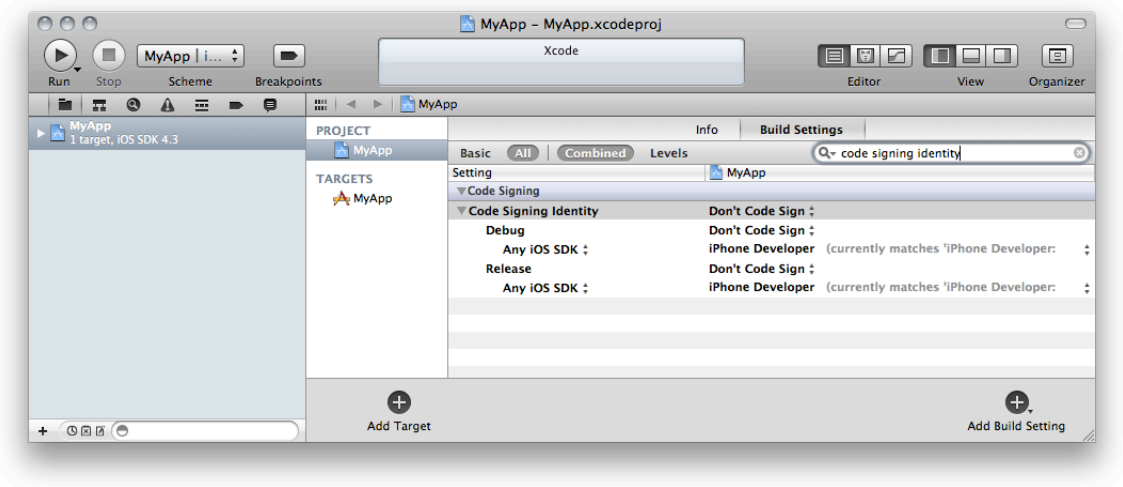
Figure 3-1 Project with a Missing SDK issue



Setting Your Code Signing Identity

When you build your app to run it on a device, Xcode signs it with a development certificate (also known as a code signing identity) stored on your keychain. To learn how to obtain and install development certificates, see "Provisioning a Device for Development" (page 10).

The Code Signing Identity build setting specifies the provisioning profile and code signing identity Xcode uses to sign your binary. Xcode looks for code signing identities in your default keychain.

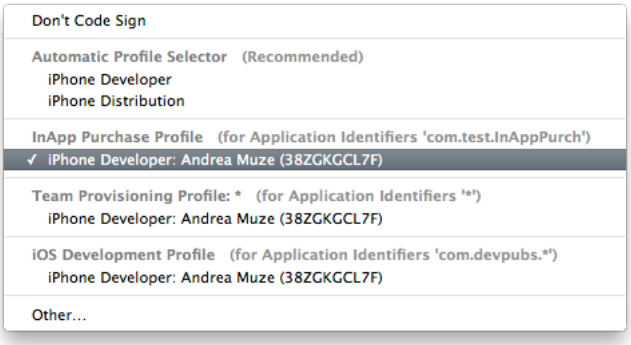


The possible values for the Code Signing Identity build setting are:

- **Don't Code Sign.** Choose this option to build only for a simulator.
- **Automatic Profile Selector.** Choose an option under this selector to select a provisioning profile whose name starts with "iPhone Developer" or "iPhone Distribution."
- **Specific Profile.** Choose the code-signing identity under a specific provisioning profile when your app requires special entitlements (see "Provisioning Your Device for Specialized Development" (page 12)). Expired or otherwise invalid provisioning profiles are dimmed and cannot be used.

  Figure 3-2 shows a set of options for the Code Signing Identity build setting with a provisioning profile for specialized development selected.

Figure 3-2 Code Signing Identity options with a specialized provisioning profile selected



Project templates are configured to use the automatic selector to set the signing identity. You need to change the value of the Code Signing Identity build setting only when your app uses a specialized provisioning profile. See "Replacing an Expired Certificate" (page 15) for details.

> **Important:** If you need to use different code signing identities that have the same name, you must use a separate Mac OS X user account for each identity.

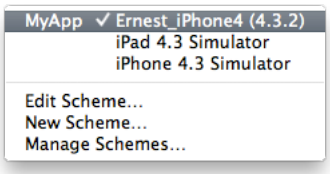
Specifying the Runtime Environment

Three aspects of your app's runtime environment are: where your app runs, what app-data is placed in its sandbox, and what location or track the Core Location framework reports to it.

Specifying the Run Destination

Before building your app, Xcode has to know where you want to run it. You specify this *run destination* in the Scheme toolbar menu. Using that menu you can switch from running your app in a simulator to running it on your device to, for example, test the device performance of your app.

When you plug into your Mac a device with a valid provisioning profile, its name and the iOS version it's running appear as an option in the Scheme toolbar menu. Use the menu to switch between running your app on a device or in a simulator.



Troubleshooting:

- "Device Not Listed as a Run Destination" (page 52).
- "Xcode Displays the "Unknown iOS Detected" Dialog When You Plug In a Device" (page 52).

Specifying the App Data

The Run action in a scheme determines the app data Xcode places in your app's sandbox before running it. Using a particular app-data archive is particularly useful in application unit tests, which check for the correct operation of critical units of code in your application. Basing your tests on a known data set allows you perform detailed unit tests without having to configure the test data in the tests themselves.

Note: Each scheme can specify only one app-data archive. If you need to use more than one app-data archive regularly, define a scheme for each archive.

Important: In Run actions, you can specify only app-data archives that that are part of your project. See "To copy app data from a development device" (page 31) for details.

To specify the app data to place in the app's sandbox before the app runs:

1. From the Scheme toolbar menu, choose the scheme you want to use.
2. Select the Run action.
3. From the Application Data pop-up menu, choose the app-data archive you want to use.

Specifying a Location or Track

To test a location-based app, you can specify a location or a track the run destination reports to the app when it launches and as it runs.

A **GPS eXhange Format (GPX)** file specifies a single location (a single waypoint) or a track (an ordered collection of waypoints). When you simulate track, the simulator or the device reports the waypoints in the order they are specified in the track.

To make a GPX file available for use in your project, add it to the project, or add a new GPX file to the project and specify the location or track details.

Building Your App

To start the build process, choose Product > Build. If the Build option is dimmed, choose a valid run destination, as described in "Specifying the Run Destination" (page 28).

The activity viewer, in the middle of the workspace window toolbar, indicates the progress of the build and whether there are build problems. If the build fails or produces warnings, you can view details about the build in the log viewer:

1. Choose View > Navigators > Log to display the log navigator.
2. In the list on the left, select the build task for which you want to view details.

   The log viewer in the editor area lists the operations that are part of the build.

If the build completes successfully, you can run your app as described in "Running Your App" (page 30).

Troubleshooting:
- "Xcode Cannot Install Your App on Your Development Device" (page 49)
- "Your Provisioning Profile Is Expired" (page 50)
- "The Code Signing Identity Build Setting Doesn't Identity a Valid Code Signing Identity in Your Keychain" (page 51)
- "Xcode Doesn't Trust Your Certificate" (page 50)
- "Your Keychain Contains Duplicate Code Signing Identities" (page 51)
- "The App ID of Your Provisioning Profile Doesn't Match Your App's Bundle Identifier" (page 52)

Running Your App

To run your app, choose Product > Run.

When you run your app, Xcode places it in a simulation environment or on a device, and launches it.

When your app runs on a device, you can ensure that it performs as you intend, using all the capabilities of your device. You should especially ensure that your app uses the device's resources—CPU, memory, battery, and so on—as efficiently as possible. See "Tuning the Performance of Your App" (page 38) for more information.

Note: To run your app on a device, the device must be connected to your Mac through a USB cable.

Managing App Data

When an app is first installed on a device or in a simulation environment, iOS creates a sandbox (also known as the app home directory) for it. As described in "File and Data Management" in *iOS App Programming Guide*, an iOS app can access only files that reside in its sandbox.

Xcode doesn't remove app data as it reinstalls an app. But you may need to erase that information as part of testing your app the way users will use it. To do so, remove the app from the Home screen. See "Uninstalling Apps" (page 34) for details.

You may also want to replace the app data in your app's sandbox with a known configuration to test specific conditions, such as when performing application unit tests. The first step is to create an archive of the app data from a development device (you cannot generate app-data archives from simulators). Then you can change the app data on your Mac and copy it back to the device.

To copy app data from a development device:

1. Plug in the device containing the app whose data you want to copy.
2. Copy the app's data from the device.

After copying the app data to the file system, you can modify it to, for example, make changes that would be tedious to do in the app itself. To access the contents of the app-data archive, Control-click the archive in the Finder and choose Show Package Contents.

If you want to use a particular configuration of your app's data when you run the app, add the appropriate app-data archive to your project, and specify that archive in the Run action of the appropriate scheme. See "Specifying the App Data" (page 28) for more information.

To copy app data to a development device:

1. Plug in the device containing the app whose data you want to replace.
2. Copy the app's data to the device.

To access your app's simulation-environment sandbox, navigate to the directory `~/Library/Application Support/iPhone Simulator/<sdk_version>/Applications` in the Finder. Then open each directory in the `Applications` directory to find your app's binary file. Alongside the binary file are the directories that make up your app's sandbox, including `Documents` and `Library`.

Further Exploration

To learn more about using Xcode to build and run apps, see "Configure Your Project".

Using iOS Simulator

You use the iOS Simulator app to run your iOS app on a Mac. By simulating the operation of your app you:

- Learn about the Xcode development experience and the iOS development environment before becoming a member of a development team.
- Find major problems in your app during design and early testing.
- Test your app's user interface.
- Measure your app's memory usage before carrying out detailed performance analysis on iOS-based devices.

The iOS Simulator app (located in `<Xcode>/Platforms/iPhoneSimulator.platform/Developer/Applications`) presents the iPhone or iPad user interface in a window on your computer. This app provides several ways to interact with it by using the keyboard and mouse to simulate taps, device rotation, and other user actions.

This chapter describes the ways in which you use your computer's input devices to simulate the interaction between users and their devices. The chapter also describes how to uninstall apps from a simulator and how to reset the contents of a simulation environment.

> **Content specifications:** This content is written for Xcode 4.3 and iOS SDK 5.0.

Setting the Device and iOS Version

iOS Simulator can simulate three devices (iPhone, iPhone with Retina display, and iPad) and several iOS versions.

To specify the device you want to simulate, choose Hardware > Device, and choose the device.

To specify the iOS version to simulate, choose Hardware > Version, and choose the iOS version.

Manipulating the Hardware

With iOS Simulator you can simulate most of the actions a user performs on a device. When your app is running in a simulator, you can carry out these hardware interactions through the Hardware menu:

- **Rotate Left.** Rotates the simulator to the left.
- **Rotate Right.** Rotates the simulator to the right.
- **Shake Gesture.** Shakes the simulator.
- **Home.** Takes the simulator to the Home screen.
- **Lock.** Locks the simulator.
- **Simulate Memory Warning.** Sends the frontmost app low-memory warnings. For information on how to handle low-memory situations, see "Observing Low-Memory Warnings" in *iOS App Programming Guide*.
- **Toggle In-Call Status Bar.** Toggles the status bar between its normal state and its state when a phone call or FaceTime call is in progress. The status bar is taller in its in-call state than in its normal state. This command shows how your app's user interface looks when the user launches your app during a call.
- **Simulate Hardware Keyboard.** Toggles the software keyboard on an iPad simulator. Turn off the software keyboard to simulate using a keyboard dock or wireless keyboard with an iPad device.
- **TV Out.** Opens a window simulating the TV out signal of a device.

Performing Gestures

Table 4-1 lists gestures you can perform on a simulator (see *iOS Human Interface Guidelines* for more about gestures).

Table 4-1 Performing gestures in iOS Simulator

Gesture	Desktop action
Tap	Click.
Touch and hold	Hold down the mouse button.
Double-tap	Double-click.
Swipe	1. Place the pointer at the place where you want the swipe to start. 2. Hold down the mouse button. 3. Move the pointer in the direction you want to swipe and release the mouse button.
Flick	1. Place the pointer at the start position. 2. Hold down the mouse button. 3. Move the pointer quickly in the direction you want to flick and release the mouse button.

Gesture	Desktop action
Drag	1. Place the pointer at the start position. 2. Hold down the mouse button. 3. Move the pointer in the direction you want to drag.
Pinch	1. Place the pointer where you want the pinch to occur. 2. Hold down the Option key. 3. Move the circles that represent finger touches to the start position. 4. Move the center of the pinch target by holding down the Shift key, moving the circles to the desired center position, and releasing the Shift key. 5. Hold down the mouse button, move the circles to the end position, and release the Option key.

Installing Apps

Xcode installs apps in simulation environments automatically when you build your app for a simulator. See "Building and Running Apps" (page 23) for details.

> **Note:** You cannot install apps from the App Store in simulation environments.

Uninstalling Apps

To uninstall apps that you have installed in a simulation environment, use the same method used to uninstall apps from devices:

1. Place the pointer over the icon of the app you want to uninstall and hold down the mouse button until the icon starts to jiggle and a close button appears.
2. Click the close button.
3. Click the Home button to stop the icon from jiggling.

Resetting Content and Settings

To set the user content and settings of a simulation environment to their factory state and remove the apps you have installed, choose iOS Simulator > Reset Content and Settings.

Viewing iOS Simulator Console and Crash Logs

To learn how to view your app's console logs when it runs in a simulator, see "Viewing Console Output and Device Logs" (page 38).

If your app crashes while running in a simulator, the CrashReporter facility displays details about the crash. You configure how CrashReporter deals with such crashes using the CrashReporterPref app, located in `<Xcode>/Applications/Utilities` (`<Xcode>`, which is the directory where the Xcode toolset is installed).

Simulation Environment File System Location

The file systems for the iOS releases the iOS Simulator can simulate are stored in your home directory, `~/Library/Application Support/iPhone Simulator`. That directory contains one subdirectory per iOS release supported by iOS Simulator.

Within each iOS-release directory, iOS Simulator stores system app preferences files in `Library/Preferences` and third-party–app preferences files in `Applications/<app_UUID>Library/Preferences`.

Hardware Simulation Support

iOS Simulator doesn't simulate accelerometer or camera hardware.